CAS Front-End Engineering

ECMASCRIPT

Silvan Gehrig

Slides provided by Michael Gfeller

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Wie entwickelt sich ECMAScript ?

- **TC39 – Ecma International, Technical Committee 39**
  - Verantwortlich für die Standardisierung vom ECMAScript
  - https://www.ecma-international.org/memento/tc39-m.htm
- **ECMA-262**
  - **ECMAScript** (or ES) is a trademarked scripting-language specification standardized by **Ecma** International in **ECMA-262**
  - JavaScript eine Implementation von ECMA-262
  - https://www.ecma-international.org/publications/standards/Ecma-262.htm
- **Ecma**
  - **Ecma International**® (original **E**uropean **C**omputer **M**anufacturers **A**ssociation) facilitates the timely creation of a wide range of global Information and Communications Technology (ICT) and Consumer Electronics (CE) standards
    - EcmaScript, C#, Eiffel, NFC
    - …
  - https://www.ecma-international.org/

ECMAScript Versionen

Version	publiziert am	Unterschiede zur Vorgängerversion
1	Juni 1997	erste Version
2	Juni 1998	Änderungen zwecks Kompatibilität zum internationalen Standard ISO/IEC 16262
3	Dezember 1999	Neu sind reguläre Ausdrücke, bessere Verarbeitung von Zeichenketten, Kontrollfluss, **Fehlerbehandlung mit try/catch**, bessere Fehlerbehandlung, bessere Formatierung bei der Ausgabe von Zahlen usw.
4	abgebrochen	Wegen Uneinigkeit in Bezug auf die Zukunft der Sprache wurde die weitere Entwicklung des komplexen Entwurfes zu ECMAScript 4 eingestellt. Einige Ideen werden in ES6 wieder aufleben.
5	Dezember 2009	Im „strict mode“ …
5.1	Juni 2011	Entspricht dem internationalen Standard ISO/IEC 16262:2011, Version 3
6 / 2015	**Juni 2015**	**Neue Syntax für komplexe Applikationen wie Klassen und Module**,
2016	**Juni 2016**	*-Operator und Array.prototype.includes
2017	**Juni 2017**	Syntactic integration with promises (**await/async**), observable streams, …
2018	**Juni 2018**	Asynchronous Iteration, Rest/Spread Properties, Regex, Promise.prototype.finally, …
2019	**Juni 2019**	Basis-Typen Erweiterungen (Array, Object, Function), …
2020	**Juni 2020**	globalThis, Optional Chaining, Nullish coalescing Operator, …

Quelle:
https://en.wikipedia.org/wiki/ECMAScript#Versions
https://en.wikipedia.org/wiki/JScript

Wie entwickelt sich EcmaScript?

- **Ab ES2015 / ES6 gibt es Jährlich eine neue Version**
  - ES2016
  - ES2017
  - ES2018
  - ES2019
  - ES2020
  - …
- **Neue Features werden neu in Stages aufgeteilt.**
  - Ziel ist ein schnellerer Release-Zyklus

Was kommt überhaupt ins neue EcmaScript

- **Features, welche nicht mit «Transpiling» gelöst werden können:**
  - Generators
  - Symbols
- **Common Patterns, welche für weitere Features benötigt werden:**
  - Promise: wird für async / await und fetch benötigt
- **Common Patterns, welche aufgrund von Performance-Gründen aufgenommen werden sollten:**
  - JSON.stringifiy

- **Zu weit abweichende Vorschläge sollten mit Libraries oder «Transpiling» implementiert werden:**
  - JSX

Prozess

- https://tc39.github.io/process-document/
- **Jedes Feature muss 4 Stages durchlaufen bevor eine Standardisierung stattfindet.**

Prozess: Stage 0

- https://github.com/tc39/proposals/blob/master/stage-0-proposals.md
- **Initial haben die neuen Vorschläge den Status «Strawperson»** (straw man)**.**
- **Jeder kann neue Features vorschlagen.**
- **Um nach Stage 1 zu erreichen muss mindestens ein Mitglied vom TC39 dieses Feature als «Champion» unterstützten.**
- **Viele Vorschläge erreichen Stage 1 nicht.**

Prozess: Stage 1

- https://github.com/tc39/proposals
- **Stage 1 Features werden «Proposal» genannt.**
- **Voraussetzungen für Stage 1:**
  - High Level API und Beispiele
  - Es muss bekannt sein, wie das Feature mit bestehenden Features interagiert.
    - Wo könnten Probleme auftauchen…
- **Es werden noch viele Änderungen erwartet.**
- **Wird vielleicht im Standard aufgenommen.**

Prozess: Stage 2

- **https://github.com/tc39/proposals**
- **Stage 2 Features werden «Draft» genannt.**
- **Voraussetzungen für Stage 2:**
  - Draft-Dokument, welches die meisten Use Cases definiert
- **Ab diesem Zeitpunkt wird erwartet, dass das Feature standardisiert wird.**
  - Das Feature wird vom TC39 Team in anderen (bestehenden) Features berücksichtig.
- **Es sollte ein Babel Plug-In vorhanden sein**
  - https://babeljs.io/docs/plugins/preset-stage-2/
- **Wird wahrscheinlich im Standard aufgenommen.**

Prozess: Stage 3

- https://github.com/tc39/proposals
- **Stage 3 Features werden «Candidate» genannt.**
- **Voraussetzungen für Stage 3:**
  - Dokument muss alle Use Cases definieren.
  - Es muss bekannt sein wie das Feature mit bestehenden Features interagiert.
  - Keine Änderungen mehr vom TC39-Team.
    - Entwicklungs-Team können Änderungen einbringen. Z.B. Chrome-Team meint das Feature ist (so) nicht umsetzbar.
  - Ein Browser muss dieses Feature (behind Flags) implementiert haben.
- **Wird sehr wahrscheinlich im Standard aufgenommen.**

Prozess: Stage 4

- **https://github.com/tc39/proposals**
- **Stage 4 Features werden «Finished» genannt.**
- **Voraussetzungen für Stage 4:**
  - Die Tests für Test262* (https://github.com/tc39/test262) müssen definiert sein.
  - Zwei Browser muss dieses Feature (behind Flags) implementiert haben und die Tests bestehen.
    - Das Feature muss über längere Zeit zum Testen verfügbar sein.
  - Keine Änderungen mehr!

- **Wird im Standard aufgenommen.**

***** Official ECMAScript Conformance Test Suite

Neue Version ES20xx

- **Jeden März werden alle Stage 4 Features in den neuen Standard aufgenommen.**
  - Verpasst? Nächstes Jahr…
- **März bis Juli werden die Texte finalisiert.**
- **EcmaScript 2020 kürzlich released!**

ES2020

- https://github.com/tc39/proposals/blob/master/finished-proposals.md

`import()`	Domenic Denicola	Domenic Denicola	June 2019	2020
`BigInt`	Daniel Ehrenberg	Daniel Ehrenberg	June 2019	2020
`Promise.allSettled`	Jason Williams Robert Pamely Mathias Bynens	Mathias Bynens	July 2019	2020
`globalThis`	Jordan Harband	Jordan Harband	October 2019	2020
`for-in` mechanics	Kevin Gibbons	Kevin Gibbons	December 2019	2020
Optional Chaining	Gabriel Isenberg Claude Pache Dustin Savery	Gabriel Isenberg Dustin Savery Justin Ridgewell Daniel Rosenwasser	December 2019	2020
Nullish coalescing Operator	Gabriel Isenberg	Gabriel Isenberg Justin Ridgewell Daniel Rosenwasser	December 2019	2020
`import.meta`	Domenic Denicola	Gus Caplan	March 2020	2020

ES20xx

- https://github.com/tc39/proposals
  - Spannende Features in der Zukunft!
  - Dieses Github-Repository entscheidet, wie sich der ES-Standard entwickelt.
    - Folgen falls interessiert!